AF
IFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:

Rogers and King

Application No. 09/742,946

Filed: December 20, 2000

For: System and Method for Performing Type Checking for Hardware Device Nodes in a Graphical Program

§ Group Art Unit: 2174
§
§ Examiner: Ke, Peng
§
§ Atty. Dkt. No.: 5150-52100
§
§
§
§
§
§
§
§
§
§
§
§

CERTIFICATE OF MAILING
37 C.F.R. § 1.8

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below:

Mark K. Brightwell
Name of Registered Representative

December 29, 2006
Date

Mark K. Brightwell
Signature

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir/Madam:

Further to the Notice of Appeal filed on September 7, 2006, Appellants present this Appeal Brief. Appellants respectfully request that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by National Instruments Corp., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 11500 N. MoPac Expwy., Austin, TX 78759-3504, as evidenced by the assignment recorded at Reel 011409, Frame 0663.

II. RELATED APPEALS AND INTERFERENCES

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-52 are pending in the application. A copy of the pending claims is included in the Claims Appendix hereto.

Claims 1-52 are rejected. The rejection of Claims 1-52 is being appealed.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been submitted subsequent to the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Claim 1 is directed to a method for propagating type information for hardware device nodes in a graphical program, as disclosed in the specification at least at the following locations:

the passage starting at page 7, line 13 and continuing through page 8, line 6;

the passage starting at page 32, line 5 and continuing through page 33, line 21;

page 37, lines 4-17; and

Figure 10.

The method operates in a computer including a display screen and a user input device. The method includes:

displaying on the display screen of the computer a first hardware device node in the graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program;

associating the first hardware device node with a hardware device;

displaying on the display screen a second hardware device node in the graphical program in response to user input;

connecting the first hardware device node to the second hardware device node in response to user input; and

propagating information from the first hardware device node to the second hardware device node, wherein the information specifies the hardware device with which the first hardware device node is associated, wherein said propagating occurs in response to said connecting the first hardware device node to the second hardware device node.

The graphical program is executable by the computer.

Claim 10 is directed to a method for performing type checking for a hardware device node in a graphical program, as disclosed in the specification at least at the following locations:

the passage starting at page 8, line 22 and continuing through page 9, line 23;

page 34, line 6 through page 35, line 17;

Figure 11;

page 37, lines 4-17; and

page 39, lines 6-19.

The method operates in a computer including a display screen. The method comprises:

displaying on the display screen of the computer a first hardware device node in the graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program;

associating the first hardware device node with a first hardware device class in response to user input;

selecting a method or property of the first hardware device class for the first hardware device node in response to user input;

changing the first hardware device node to have an association with a second hardware device class in response to user input; and

performing type checking to determine whether the method or property is valid for the second hardware device class, in response to said changing the first hardware device node to have an association with the second hardware device class.

The graphical program is executable by the computer.

Claim 18 is directed to a memory medium comprising program instructions. Claim 18 is supported in the specification similarly to the method of Claim 1 described above. In addition, see the passage starting at page 17, line 16 and continuing through page 18, line 11. The program instructions are executable to:

display a first hardware device node in a graphical program in response to user

input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program;

associate the first hardware device node with a hardware device;

display on a display screen a second hardware device node in the graphical program in response to user input;

connect the first hardware device node to the second hardware device node in response to user input; and

propagate information from the first hardware device node to the second hardware device node, wherein the information specifies the hardware device with which the first hardware device node is associated, wherein said propagating occurs in response to said connecting the first hardware device node to the second hardware device node.

The graphical program is executable by a computer system.

Claim 26 is directed to a memory medium comprising program instructions. Claim 26 is supported in the specification similarly to Claim 10 described above. In addition, see the passage starting at page 17, line 16 and continuing through page 18, line 11. The program instructions are executable to:

display a first hardware device node in a graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program;

associate the first hardware device node with a first hardware device class in response to user input;

select a method or property of the first hardware device class for the first hardware

device node in response to user input;

change the first hardware device node to have an association with a second hardware device class in response to user input; and

perform type checking to determine whether the method or property is valid for the second hardware device class, in response to said changing the first hardware device node to have an association with the second hardware device class.

The graphical program is executable by a computer system.

Claim 32 is directed to a system for propagating type information for hardware device nodes in a graphical program. Claim 32 is supported in the specification similarly to Claim 1 described above. In addition, see the passage starting at page 18, line 14 and continuing through page 19, line 23. The system comprises: a computer including a processor coupled to a memory; a display screen coupled to the computer; and a user input device coupled to the computer. The processor is operable to execute program instructions stored in the memory to:

display on the display screen a first hardware device node in the graphical program in response to user input received from the user input device, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the program;

associate the first hardware device node with a hardware device;

display on the display screen a second hardware device node in the graphical program in response to user input received from the user input device;

connect the first hardware device node to the second hardware device node in response to user input received from the user input device; and

propagate information from the first hardware device node to the second hardware

device node, wherein the information specifies the hardware device with which the first hardware device node is associated, wherein said propagating occurs in response to said connecting the first hardware device node to the second hardware device node.

The graphical program is executable by the computer.

Claim 39 is directed to a system for performing type checking for a hardware device node in a graphical program. Claim 39 is supported in the specification similarly to Claim 10 described above. In addition, see the passage starting at page 18, line 14 and continuing through page 19, line 23. The system comprises: a computer including a processor coupled to a memory; and a display screen coupled to the computer. The processor is operable to execute program instructions stored in the memory to:

display on the display screen a first hardware device node in the graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program;

associate the first hardware device node with a first hardware device class in response to user input;

select a method or property of the first hardware device class for the first hardware device node in response to user input;

change the first hardware device node to have an association with a second hardware device class in response to user input; and

perform type checking to determine whether the method or property is valid for the second hardware device class, in response to said changing the first hardware device node to have an association with the second hardware device class.

The graphical program is executable by the computer.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-14, 16-29 and 31-52 were rejected under 35 U.S.C. Section 102(e) as being anticipated by Teranishi et al. (USPN 6,117,183) (hereinafter referred to simply as "Teranishi").

Claims 15 and 30 were rejected under 35 U.S.C. Section 103(a) as being unpatentable over Teranishi in view of McKaskle et al. (USPN 5,481,741) (hereinafter referred to simply as "McKaskle").

VII. ARGUMENT

First Ground of Rejection

Claims 1-14, 16-29 and 31-52 were rejected under 35 U.S.C. Section 102(e) as being anticipated by Teranishi et al. (USPN 6,117,183) (hereinafter referred to simply as "Teranishi"). Appellants traverse this rejection based on the following reasoning. Different groups of claims are addressed under their respective subheadings.

Claims 1, 18 and 32

Teranishi never suggests propagating information from a first hardware device node to a second hardware device node *of a graphical program*, where the information specifies the hardware device with which the first hardware device node is associated. Regarding this feature, the Examiner states on page 14 of the Final Action:

> "Teranishi teaches this limitation because Teranishi locates the problematic sign[al] path within a circuit. (column 4, lines 1-25) In order the [sic] locate the problematic path, Teranishi simulates the circuit (column 6, lines 17-30) and its components and that requires propagating information from one hardware device node to another device. (column 6, lines 46-58, column 8, lines 12-53)"

Applicant respectfully disagrees. Teranishi teaches extracting (i.e., identifying) a signal path that does not satisfy a target cycle time as an error path based on signal path data (SPD) 44, where the signal path data contains a delay value calculated for each signal path from the component placement data (Col. 4, lines 1-5). This extraction process does not in any way suggest the existence or use of nodes in a graphical program. Thus, it is not possible that the extraction process could suggest propagating information from a first hardware device node to a second hardware device node, where the information specifies the hardware device with which the first hardware device node is associated as recited in Claims 1, 18 and 32.

Furthermore, Teranishi never suggests the idea of a node in a graphical program being associated with a hardware device.

In the passage quoted above, the Examiner states that

> "... Teranishi simulates the circuit (column 6, lines 17-30) and its components and that requires propagating information from one hardware device node to another device". [*Emphasis added*].

Applicant notes that Teranishi never suggests simulating the functional behavior of the displayed elements (such as elements L0 through L12 of Figure 2). Teranishi appears to be primarily concerned with the computation and display of time delays associated with worst-case paths in the component placement diagram in order to enable the user to move components in a way that decreases the worst-case time delays. (See Col. 4, lines 6-58 and especially Figure 3.) Thus, the Examiner's reasoning is based on the faulty premise that Teranishi teaches simulating a circuit indicated by the component placement diagram.

In addition, even if one were to assume *arguendo* that Teranishi did teach simulating a circuit indicated by the component placement diagram, it does not follow that this teaching would require the propagation of information specifying a hardware device from one hardware device node to another hardware device node in a graphical program. The Examiner is improperly importing teachings from the Applicant's specification into

Teranishi.

Each of independent claims 1, 18 and 32 recites something similar to:

> "displaying a first hardware device node in a graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program". [*Underlining added*]

Applicant's specification defines the term "graphical program" in part at page 3, lines 17-27:

> "In response to the user constructing a diagram or graphical program using the block diagram editor, data structures may be automatically constructed which characterize an execution procedure which corresponds to the displayed procedure. The graphical program may be compiled or interpreted by a computer." [*Underlining added*]

> "Therefore, Kodosky et al teaches a graphical programming environment wherein a user places or manipulates icons and interconnects or "wires up" the icons in a block diagram using a block diagram editor to create a graphical "program." . . . Thus, a user can create a computer program solely by using a graphically based programming environment." [*Underlining added*]

Teranishi never suggests anything close to a graphical program or to a node in a graphical program. The component placement diagram of Figure 2 of Teranishi is not a graphical program as recited in the claims. Teranishi never suggests that the displayed items L0 through L12 and their interconnections visually indicate the functionality of a program. Furthermore, there is no sense in which the component placement diagram defines a program which can be compiled or interpreted by a computer.

On page 14 of the Final Action, the Examiner states that "Teranishi teaches a graphical program because Teranishi discloses a graphical user interface that allows a user to move, update, and associate virtual objects. (column 4, lines 51-68)". It appears that the Examiner is confusing a graphical user interface that allows a user to manipulate objects on a screen with a graphical program. A graphical program is a program whose functionality is visually indicated by the plurality of interconnected nodes or icons as

recited in the claims. Teranishi nowhere suggests a graphical program or anything related to graphical programming.

Therefore, Claims 1, 18 and 32, and their dependents, are patentably distinguished over Teranishi.

Claims 3, 20 and 24

Each of claims 3, 20 and 24 recites a feature similar to the feature of "associating the second hardware device node with the hardware device with which the first hardware device node is associated, in response to said propagating the information to the second hardware device node." There is no sense in which Teranishi teaches two entities being associated with the same hardware device. Thus, it is not possible that Teranishi should teach or suggest this feature. The Examiner points to Teranishi Col. 4, lines 5-26 as evidence for the anticipation of this feature. Col. 4, lines 5-16 read as follows:

> If an error path exists, a table 51 of an error path list (EPL) is constructed in memory under the direction of an operator (designer) (steps 106, 108). This list is constructed by sorting error paths in order of decreasing delay time or by giving priority to paths associated with a designated component. The reason is that when the device increases in size or complexity, a large number of error paths may be detected (especially, in verification at early stages of the design), and in such cases, the amount of listed contents will become enormous, making it difficult to locate problematic signal paths. The list is displayed in a window on the display screen. For example, as shown in the screen display example of FIG. 2, the error path list is displayed with its contents arranged in order of decreasing delay time in a window W2 different from the window W1 where the component placement diagram is displayed. The illustrated example shows the result of verification done based on a cycle time of 2000.00 ps. Of the seven paths listed, six error paths exceeding the cycle time are detected and labelled an error mark E. Here, it is apparent that the number of signal paths displayed is limited by the size of the window.

This passage has absolutely nothing to do with associating a second hardware device node of a graphical program with a hardware device that is associated with the first hardware device node. Thus, Claims 3, 20 and 24 are further distinguished over Teranishi.

Claims 10, 26 and 39

Each of Claims 10, 26 and 39 recites features similar to the following:

> "associating the first hardware device node with a first hardware device class in response to user input";
>
> "selecting a method or property of the first hardware device class for the first hardware device node in response to user input";
>
> "changing the first hardware device node to have an association with a second hardware device class in response to user input"; and
>
> "performing type checking to determine whether the method or property is valid for the second hardware device class, in response to said changing the first hardware device node to have an association with the second hardware device class".

Teranishi never suggests any of these features. The notion of hardware device class is entirely missing from Teranishi. Thus, it is not possible to reasonably construe Teranishi as teaching the operation of associating a hardware device node with a hardware device class, or, the operation of selecting a method or property of a hardware device class as recited in Claims 10, 26 and 39.

The Examiner points to Column 7, lines 40-45 of Teranishi as providing evidence for the anticipation of the "associating the first hardware device node with a first hardware device class in response to user input". Column 7, lines 40-45 read as follows:

> A mesh pass count calculation routine 33 calculates the number of interconnection routes of the associated paths that pass across each interconnection mesh. Here, the interconnection mesh is one of the sections into which the entire routing region is divided horizontally and vertically, as shown by M1, M2, and M3 in Figure 7. [*Underlining added*]

This passage has nothing to do with associating a hardware device node in a graphical program with a hardware device class as recited in claims 10, 26 and 39. This passage uses the term "associated paths". However, it contains absolutely no suggestion of associating a node of a graphical program with anything.

Each of independent claims 10, 26 and 39 recites something similar to:

"displaying on the display screen of the computer a first hardware device node in the graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program".

Teranishi never suggests anything close to a graphical program or to a node in a graphical program. In this regard, see the arguments presented above relative to claims 1, 18 and 32.

Therefore, Claims 10, 26 and 39, and their dependents are patentably distinguished over the Teranishi at least for the reasons given above.

Claims 16, 31 and 40

Each of Claims 16, 31 and 40 recites a feature similar to the following: "wherein said performing type checking to determine whether the method or property is valid for the second hardware device class comprises: determining a list of valid methods and properties for the second hardware device class; and determining whether the method or property is included in the list of valid method and properties." The Examiner point to the passage at Col. 6, line 58 to Col. 7, line 10 of Teranishi as evidence for the anticipation of this feature. This passage reads as follows:

> An interconnection display routine 30 displays the interconnections of the associated signal paths on the component placement diagram. The interconnections to be displayed may be represented by straight lines interconnecting the terminals of the components, as shown in FIG. 4, for example, or alternatively, the interconnections may be displayed along the actually routed lines. When displaying the interconnections, the interconnecting lines shown in the component placement diagram are displayed in different colors for different error levels. For example, an error path is displayed in red and an improved path in green; it is also possible to classify the signal paths according to their values expressed in percentage of a verification reference value and to display a path with a value exceeding 150% in red, 150 to 100% in pink, 100 to 80% in yellow, and less than 80% no display, for example. Alternatively, the line thickness or line type may be changed to display different error levels.

This passage has nothing to do with *determining a list of valid methods and properties*

for a hardware device class. Thus, it is not reasonable to suggest that it teaches "determining whether the method or property is included in the list of valid method and properties". Therefore, Claims 16, 31 and 40 are additionally distinguished over Teranishi.

Second Ground of Rejection

Claims 15 and 30 were rejected under 35 U.S.C. Section 103(a) as being unpatentable over Teranishi in view of McKaskle et al. (USPN 5,481,741) (hereinafter referred to simply as "McKaskle"). Appellants traverse this rejection based on the following reasoning.

The Examiner relies on Figures 100A-D of McKaskle to teach "where the first hardware device node is a register access node" as recited in each of Claims 15 and 30. Regarding Figures 100A-D, at Column 49, lines 60-67, McKaskle teaches:

> Referring now to FIG. 100, to initialize the shift register with a specific value, the initial value is wired to the left terminal of the shift register (outside the While Loop). If the initial value is left unwired, the initial value will be the default value for the shift register data type. For example, if the shift register data type is Boolean, the initial value will be FALSE. Similarly, if the shift register data type is numeric, the initial value will be zero.
>
> It is noted that values stored in the shift register are not discarded until the VI is closed and removed from memory. In other words, if a VI is run which contains uninitialized shift registers, the initial values for the subsequent run will be the ones left from the previous run.

Thus, it appears that Figures 100A-D are referring to a construct that performs the function of a shift register. In contrast, a register access node is a node that is used to access a register (or registers) of a hardware device, as taught in the Applicant's specification, e.g., at page 10, lines 1-4:

> In one embodiment, a register access node or primitive may be included in a graphical program, such that when the graphical program is executed on the computer system, the register access node is operable to access (i.e., either read or write) registers of a hardware device in the computer system.

Thus, it appears that Figures 100A-D of McKaskle are not referring to a register access node.

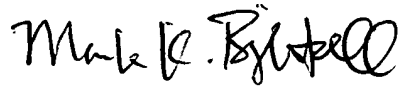
Therefore, Claims 15 and 30 are additionally distinguished over Teranishi and McKaskle.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-52 was erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5150-52100/JCH. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,



Mark K. Brightwell
Reg. No. 47,446
Agent for Appellant

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8800

Date: December 29, 2006

IX. CLAIMS APPENDIX

The claims on appeal are as follows.

1. (Previously Presented) A method for propagating type information for hardware device nodes in a graphical program, wherein the method operates in a computer including a display screen and a user input device, the method comprising:

displaying on the display screen of the computer a first hardware device node in the graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program;

associating the first hardware device node with a hardware device;

displaying on the display screen a second hardware device node in the graphical program in response to user input;

connecting the first hardware device node to the second hardware device node in response to user input;

propagating information from the first hardware device node to the second hardware device node, wherein the information specifies the hardware device with which the first hardware device node is associated, wherein said propagating occurs in response to said connecting the first hardware device node to the second hardware device node;

wherein the graphical program is executable by the computer.

2. (Original) The method of claim 1,

wherein said displaying the first and second hardware device nodes in the graphical program comprises including the first and second hardware device nodes in a block diagram of the graphical program, wherein the block diagram visually indicates functionality of the graphical program.

3. (Original) The method of claim 1, further comprising:

associating the second hardware device node with the hardware device with which the first hardware device node is associated, in response to said propagating the information to the second hardware device node.

4. (Original) The method of claim 1,

wherein said connecting the first hardware device node to the second hardware device node comprises connecting a wire from an output terminal of the first hardware device node to an input terminal of the second hardware device node.

5. (Original) The method of claim 1,

wherein said associating the first hardware device node with a hardware device comprises associating the first hardware device node with a hardware device class corresponding to the hardware device;

wherein said propagating information from the first hardware device node to the second hardware device node comprises propagating information specifying the hardware device class with which the first hardware device node is associated.

6. (Original) The method of claim 5, further comprising:

associating the second hardware device node with the hardware device class, in response to said propagating the information to the second hardware device node.

7. (Original) The method of claim 6, further comprising:

associating the second hardware device node with a method of the hardware device class in response to user input;

wherein during execution of the graphical program the second hardware device node is operable to invoke the method.

8. (Original) The method of claim 6, further comprising:

associating the second hardware device node with a property of the hardware device class in response to user input;

wherein during execution of the graphical program the second hardware device node is operable to perform one or more of: 1) getting the property; and 2) setting the property.

9. (Original) The method of claim 1, further comprising:

executing the graphical program, wherein during execution of the graphical program the second hardware device node is operable to access the hardware device.

10. (Previously Presented) A method for performing type checking for a hardware device node in a graphical program, wherein the method operates in a computer including a display screen, the method comprising:

displaying on the display screen of the computer a first hardware device node in the graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program;

associating the first hardware device node with a first hardware device class in response to user input;

selecting a method or property of the first hardware device class for the first hardware device node in response to user input;

changing the first hardware device node to have an association with a second hardware device class in response to user input; and

performing type checking to determine whether the method or property is valid for the second hardware device class, in response to said changing the first hardware device node to have an association with the second hardware device class;

wherein the graphical program is executable by the computer.

11. (Original) The method of claim 10, further comprising:

indicating an invalid condition if the method or property is not valid for the second hardware device class.

12. (Original) The method of claim 11,

wherein said indicating the invalid condition comprises altering the visual appearance of a wire connected to an input terminal of the first hardware device node, wherein the wire provides information specifying the second hardware device class with which the first hardware device node is associated.

13. (Original) The method of claim 10, further comprising:

preventing execution of the graphical program if the method or property is not valid for the second hardware device class.

14. (Original) The method of claim 10,

wherein the first hardware device node has an input terminal for receiving information specifying a hardware device class with which to associate the first hardware device node;

wherein said associating the first hardware device node with the first hardware device class comprises connecting a first wire to the input terminal;

wherein said changing the first hardware device node to have an association with a second hardware device class comprises connecting a second wire to the input terminal.

15. (Original) The method of claim 10,

wherein the first hardware device node is a register access node.

16. (Original) The method of claim 10,

wherein said performing type checking to determine whether the method or property is valid for the second hardware device class comprises:

determining a list of valid methods and properties for the second hardware device class; and

determining whether the method or property is included in the list of valid method and properties.

17. (Original) The method of claim 16,

wherein said determining the list of valid methods and properties for the second hardware device class comprises determining the valid methods and properties from a type library, wherein the type library includes information regarding the second hardware device class.

18. (Previously Presented) A memory medium comprising program instructions executable to:

display a first hardware device node in a graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program;

associate the first hardware device node with a hardware device;

display on a display screen a second hardware device node in the graphical program in response to user input;

connect the first hardware device node to the second hardware device node in response to user input;

propagate information from the first hardware device node to the second hardware device node, wherein the information specifies the hardware device with which the first hardware device node is associated, wherein said propagating occurs in response to said connecting the first hardware device node to the second hardware device node;

wherein the graphical program is executable by a computer system.

19. (Original) The memory medium of claim 18,

wherein said displaying the first and second hardware device nodes in the graphical program comprises including the first and second hardware device nodes in a

block diagram of the graphical program, wherein the block diagram visually indicates functionality of the graphical program.

20. (Original) The memory medium of claim 18, further comprising program instructions executable to:

associate the second hardware device node with the hardware device with which the first hardware device node is associated, in response to said propagating the information to the second hardware device node.

21. (Original) The memory medium of claim 18,

wherein said connecting the first hardware device node to the second hardware device node comprises connecting a wire from an output terminal of the first hardware device node to an input terminal of the second hardware device node.

22. (Original) The memory medium of claim 18,

wherein said associating the first hardware device node with a hardware device comprises associating the first hardware device node with a hardware device class corresponding to the hardware device;

wherein said propagating information from the first hardware device node to the second hardware device node comprises propagating information specifying the hardware device class with which the first hardware device node is associated.

23. (Original) The memory medium of claim 22, further comprising program instructions executable to:

associate the second hardware device node with the hardware device class, in response to said propagating the information to the second hardware device node.

24. (Original) The memory medium of claim 23, further comprising program instructions executable to:

associate the second hardware device node with a method of the hardware device class in response to user input;

wherein during execution of the graphical program the second hardware device node is operable to invoke the method.

25. (Original) The memory medium of claim 23, further comprising program instructions executable to:

associate the second hardware device node with a property of the hardware device class in response to user input;

wherein during execution of the graphical program the second hardware device node is operable to perform one or more of: 1) getting the property; and 2) setting the property.

26. (Previously Presented) A memory medium comprising program instructions executable to:

display a first hardware device node in a graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program;

associate the first hardware device node with a first hardware device class in response to user input;

select a method or property of the first hardware device class for the first hardware device node in response to user input;

change the first hardware device node to have an association with a second hardware device class in response to user input; and

perform type checking to determine whether the method or property is valid for the second hardware device class, in response to said changing the first hardware device node to have an association with the second hardware device class;

wherein the graphical program is executable by a computer system.

27. (Original) The memory medium of claim 26, further comprising program instructions executable to:

indicate an invalid condition if the method or property is not valid for the second hardware device class.

28. (Original) The memory medium of claim 26, further comprising program instructions executable to:

prevent execution of the graphical program if the method or property is not valid for the second hardware device class.

29. (Original) The memory medium of claim 26,

wherein the first hardware device node has an input terminal for receiving information specifying a hardware device class with which to associate the first hardware device node;

wherein said associating the first hardware device node with the first hardware device class comprises connecting a first wire to the input terminal in response to user input;

wherein said changing the first hardware device node to have an association with a second hardware device class comprises connecting a second wire to the input terminal in response to user input.

30. (Original) The memory medium of claim 26,

wherein the first hardware device node is a register access node.

31. (Original) The memory medium of claim 26,

wherein said performing type checking to determine whether the method or property is valid for the second hardware device class comprises:

determining a list of valid methods and properties for the second hardware device class; and

determining whether the method or property is included in the list of valid method and properties.

32. (Previously Presented) A system for propagating type information for hardware device nodes in a graphical program, the system comprising:

a computer including a processor coupled to a memory;

a display screen coupled to the computer;

a user input device coupled to the computer;

wherein the processor is operable to execute program instructions stored in the memory to:

display on the display screen a first hardware device node in the graphical program in response to user input received from the user input device, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the program;

associate the first hardware device node with a hardware device;

display on the display screen a second hardware device node in the graphical program in response to user input received from the user input device;

connect the first hardware device node to the second hardware device node in response to user input received from the user input device;

propagate information from the first hardware device node to the second hardware device node, wherein the information specifies the hardware device with which the first hardware device node is associated, wherein said propagating occurs in response to said connecting the first hardware device node to the second hardware device node;

wherein the graphical program is executable by the computer.

33. (Original) The system of claim 32,

wherein said displaying the first and second hardware device nodes in the graphical program comprises including the first and second hardware device nodes in a block diagram of the graphical program, wherein the block diagram visually indicates functionality of the graphical program.

34. (Original) The system of claim 32, wherein the processor is further operable to execute program instructions stored in the memory to:

associate the second hardware device node with the hardware device with which the first hardware device node is associated, in response to said propagating the information to the second hardware device node.

35. (Original) The system of claim 32,

wherein said associating the first hardware device node with a hardware device comprises associating the first hardware device node with a hardware device class corresponding to the hardware device;

wherein said propagating information from the first hardware device node to the second hardware device node comprises propagating information specifying the hardware device class with which the first hardware device node is associated.

36. (Original) The system of claim 35,

wherein the processor is further operable to execute program instructions stored in the memory to associate the second hardware device node with the hardware device class, in response to said propagating the information to the second hardware device node.

37. (Original) The system of claim 36, wherein the processor is further operable to execute program instructions stored in the memory to:

associate the second hardware device node with a method of the hardware device class in response to user input;

wherein during execution of the graphical program the second hardware device node is operable to invoke the method.

38. (Original) The system of claim 36, wherein the processor is further operable to execute program instructions stored in the memory to:

associate the second hardware device node with a property of the hardware device class in response to user input;

wherein during execution of the graphical program the second hardware device node is operable to perform one or more of: 1) getting the property; and 2) setting the property.

39. (Previously Presented) A system for performing type checking for a hardware device node in a graphical program, the system comprising:

a computer including a processor coupled to a memory;

a display screen coupled to the computer;

wherein the processor is operable to execute program instructions stored in the memory to:

display on the display screen a first hardware device node in the graphical program in response to user input, wherein the graphical program comprises a plurality of interconnected nodes or icons, wherein the plurality of interconnected nodes or icons visually indicate functionality of the graphical program;

associate the first hardware device node with a first hardware device class in response to user input;

select a method or property of the first hardware device class for the first hardware device node in response to user input;

change the first hardware device node to have an association with a second hardware device class in response to user input; and

perform type checking to determine whether the method or property is valid for the second hardware device class, in response to said changing the first hardware device node to have an association with the second hardware device class;

wherein the graphical program is executable by the computer.

40. (Original) The system of claim 39,

wherein said performing type checking to determine whether the method or property is valid for the second hardware device class comprises:

determining a list of valid methods and properties for the second hardware device class; and

determining whether the method or property is included in the list of valid method and properties.

41. (Previously Presented) The system of claim 39, wherein the graphical program is interpretable or compilable to generate instructions executable by the computer.

42. (Previously Presented) The system of claim 39, wherein the graphical program comprises a dataflow diagram.

43. (Previously Presented) The system of claim 32, wherein the graphical program is interpretable or compilable to generate instructions executable by the computer.

44. (Previously Presented) The system of claim 32, wherein the graphical program comprises a dataflow diagram.

45. (Previously Presented) The memory medium of claim 26, wherein the graphical program is interpretable or compilable to generate instructions executable by the computer system.

46. (Previously Presented) The memory medium of claim 26, wherein the graphical program comprises a dataflow diagram.

47. (Previously Presented) The memory medium of claim 18, wherein the graphical program is interpretable or compilable to generate instructions executable by the computer system.

48. (Previously Presented) The memory medium of claim 18, wherein the graphical program comprises a dataflow diagram.

49. (Previously Presented) The method of claim 10, wherein the graphical program is interpretable or compilable to generate instructions executable by the computer.

50. (Previously Presented) The method of claim 10, wherein the graphical program comprises a dataflow diagram.

51. (Previously Presented) The method of claim 1, wherein the graphical program is interpretable or compilable to generate instructions executable by the computer.

52. (Previously Presented) The method of claim 1, wherein the graphical program comprises a dataflow diagram.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.